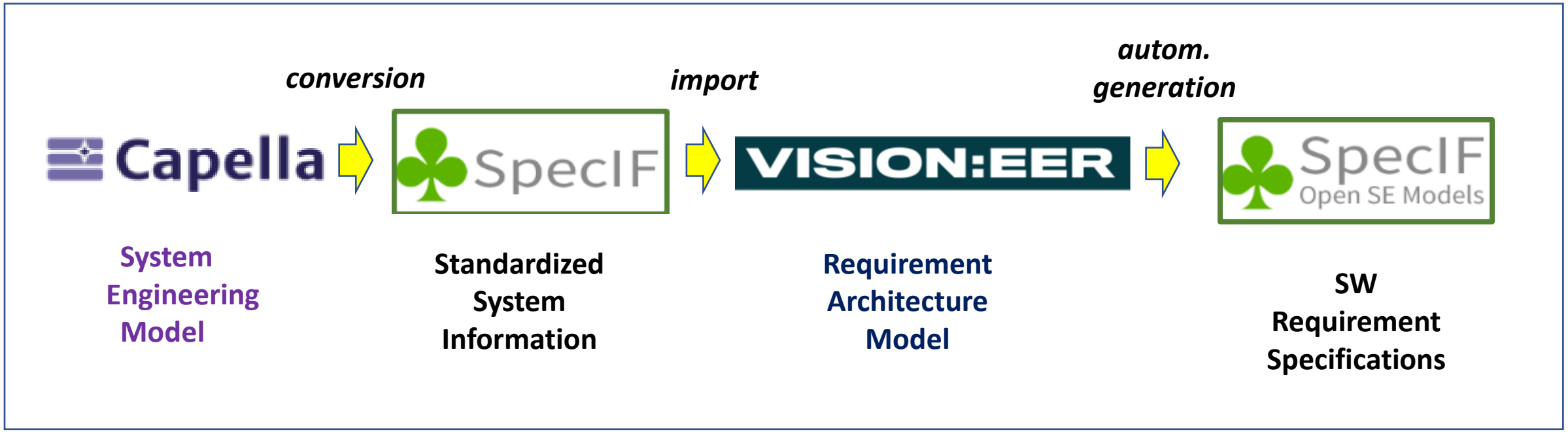
Example RML classes for Demo Project



VISION:EER

Contents of Demo Project

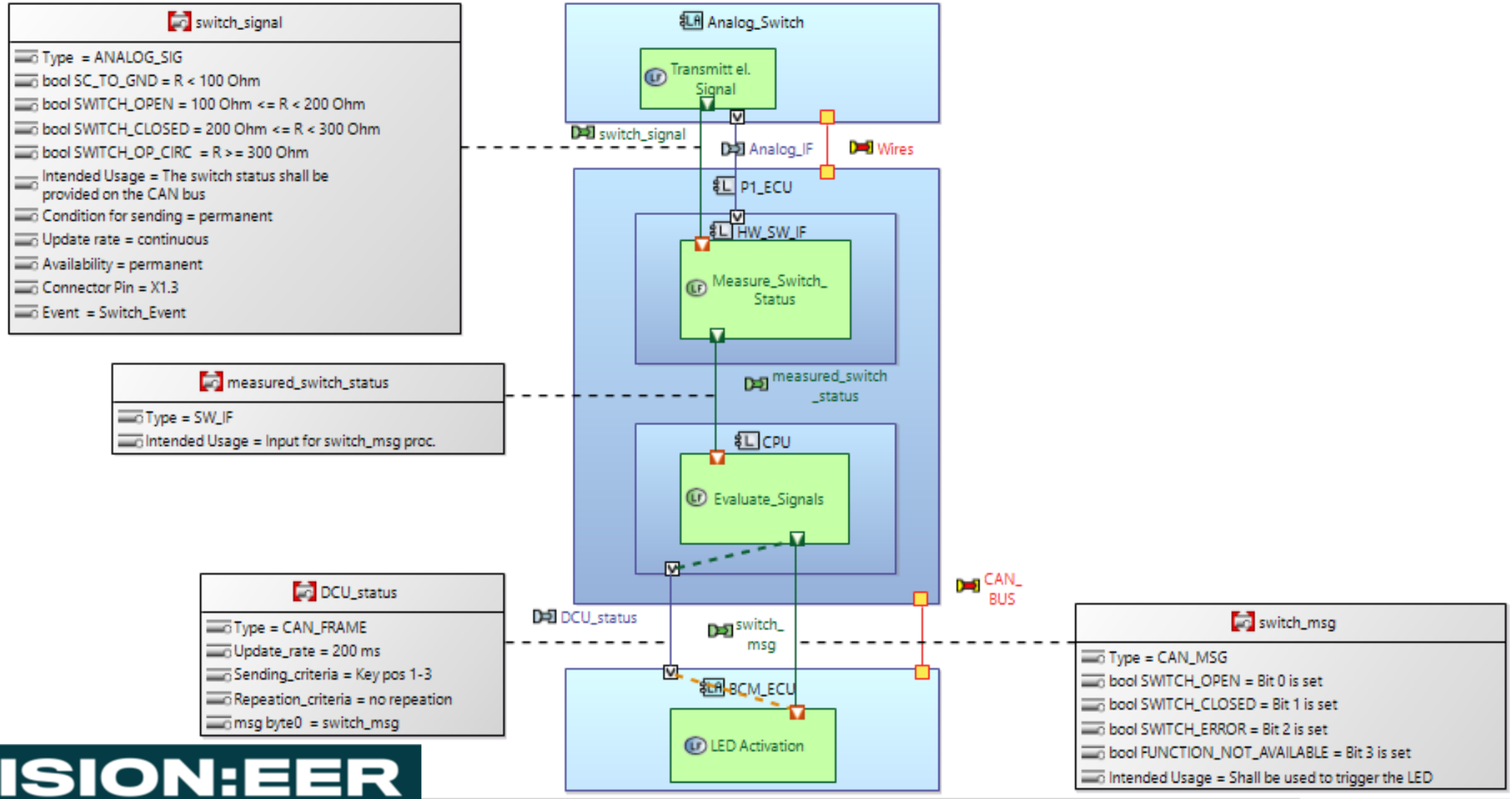
The following presentation shows:

- **How the system model information *(=System Demands)* is imported into RML**
- **How it is merged with the generic requirements for embedded system *(=Subsystem Solutions)* of the system kit**
- **How they HW architecture related requirements (*=Library)* are inherited from platform++**

→ **The tasks of the requirement architect how to create complete (but "blank") specification structures containing those information**

VISION:EER

Capella System Model: P1_ECU



VISION:EER

Requirememt Architecture Tasks

The Requirement Architect has to

- Build a RML model and tailor the generic RML classes according to the project specific needs
- Goal: Generation of two specifications in Specif-format automatically out of the RML model
  - **SRS_P1_ECU** → containing all ECU specific SW requirements
  - **SRS_P1_HSI** → containing all HW/SW interface requirements

  (see SRS_P1_ECU.pdf and SRS_P1_HSI.pdf)

→ **Note: Only the red text in the following classes must be added to achieve that goal**

VISION:EER

Imported System Information from Capella

Imported Signals:

```
REQ_Block   switch_signal extends SIGNAL {
   struc     Type_and_direction = AN_IN;
   struc     Sender = Analog_Switch;
   struc     Receiver = Measure_Switch_Status;
   p_goal    Intended_usage = "The switch status shall be provided on the
                                CAN bus";
   req       Update_rate = "continuous";
   req       Condition_for_sending = "permanent";
   bool      SWITCH_OPEN = "100 Ohm <= R < 200 Ohm";
   bool      SWITCH_CLOSED = "200 Ohm <= R < 300 Ohm";
   bool      SWITCH_SC_GND = "R < 100 Ohm";
   bool      SWITCH_OP_CIRC = "R >= 300 Ohm";
   struc     Connector_pin = " X1.3";
   REQ_Block      Event = Switch_Event ; }
```

Imported Events:

```
IF_REQ   Switch_Event extends EVENT {
   req        Ev_qualific_crit;
   p_goal     Event_actions;
   p_goal     Ev_reoccur_handle;
   req        Ev_reaction_time;
   req        (0..1)Ev_traceability_action; }
```

Imported SW_IFs :

```
REQ_Block   measured_switch_status extends SW_IF {
   struc     Type_and_direction = SW_IF;
   struc     Sender = Measure_Switch_Status;
   struc     Receiver = Evaluate Signals;
   p_goal    Intended_usage = "Input for switch_msg proc.";
   req       Update_rate = "Update_rate_of _SF()";
   bool      (0..*) xx_STATE;
   value     (0..1) xx_VALUE
   req       Default_Value; }
```

Imported Frames:

```
REQ_Block   DCU_status  extends  FRAME {
   struc    Type_and_direction = CAN_OUT;
   struc    Sender = Evaluate_Signals;
   struc    Receiver = BCM_ECU;
   req      Update_rate = "200 ms";
   req      Frame_sending_criteria = "Key_Pos 1-3";
   req      Frame_repetion_criteria = "no repeation";
   msg      Message= "byte 0 -> switch_msg";}
```

Imported Messages:

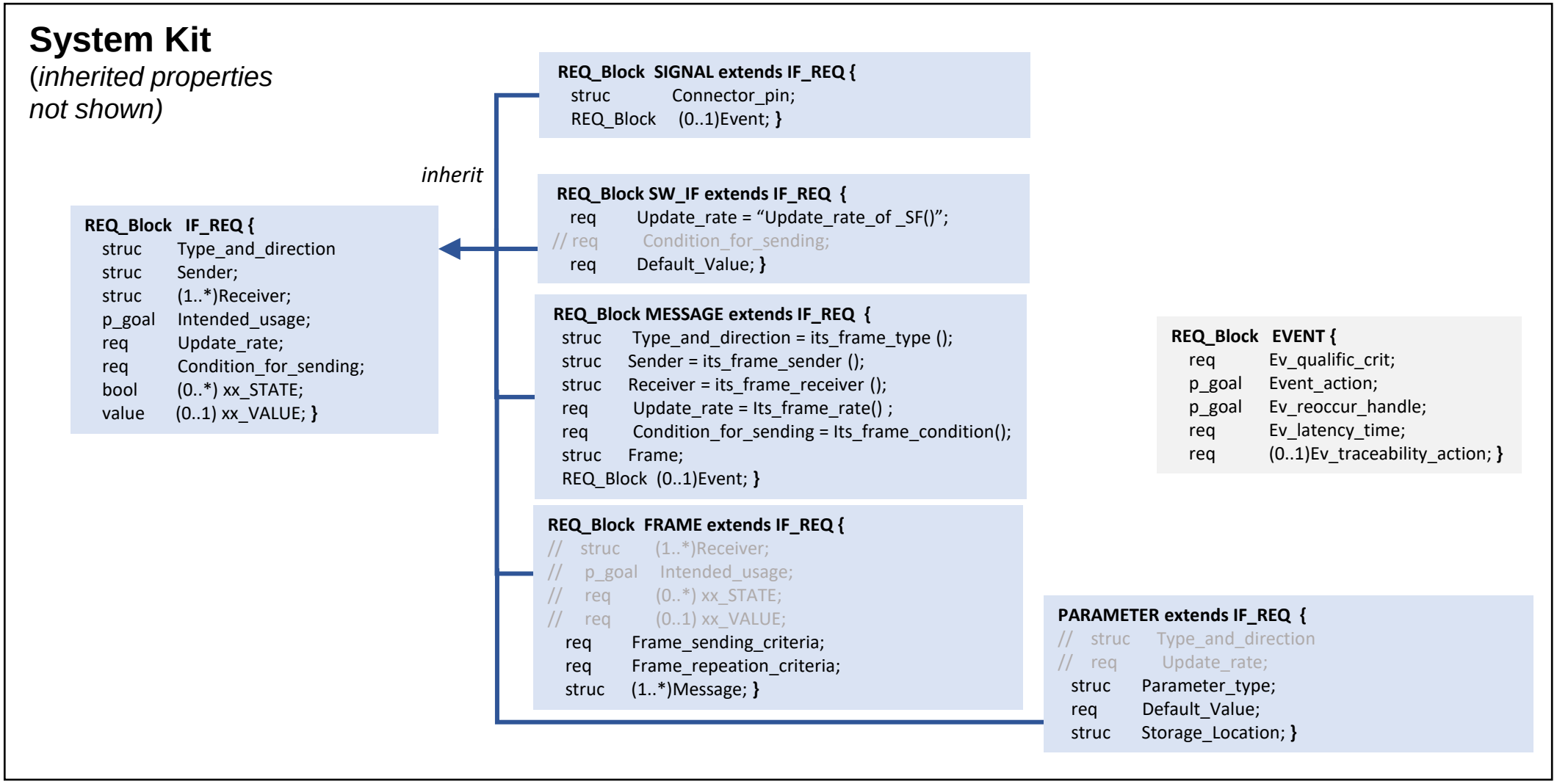
```
REQ_Block   switch_msg extends MESSAGE {
   struc     Type_and_direction = its_frame_type ();
   struc     Sender = its_frame_sender ();
   struc     Receiver = its_frame_receiver ();
   preq      Intended_usage = "Shall be used to trigger the LED";
   bool      SWITCH_OPEN = "Bit 0 is set";
   bool      SWITCH_CLOSED = "Bit 1 is set";
   bool      SWITCH_ERROR = "Bit 2 is set";
   bool      FUNCTION_NOT_AVAILABLE = "Bit 3 is set";
   struc     Frame = "DCU_status";
   struc     Event = "no"; }
```

Imported System Functions :

```
REQ_Block   P1_ECU {
   REQ_Block   SF = Measure_Switch_Status;
   REQ_Block   SF = Evaluate_Signals; }
```

VISION:EER

Generic Interface Requirements



Merging of CAPELLA System Functions → Spec SRS_P1_ECU

Capella System Model:

```
REQ_Block   P1_ECU {
   REQ_Block   SF = Measure_Switch_Status;
   REQ_Block   SF = Evaluate_Signals; }
```

merge

```
Conversion Result
REQ_Block   = P1_System_Functions  merge P1_ECU and SFs {
    REQ_Block    SF = Measure_Switch_Status_SF;
    REQ_Block    SF = Evaluate_Signals_SF;  }
```

merge

```
System Kit
REQ_Block   SFs {
    REQ_Block     (0..*) SF = xx extends SF;  }
```

```
System Kit
REQ_Block   ECU {
   MTHD          Black_Box_Interfaces = its_BB_IFs();
   MTHD          Event = its_Events();
   REQ_Block     System_Functions;
   REQ_Block     Supplement_Functions = XX_SUP;  }
```

```
System Kit
SPEC              Spec_template {
   INFO          Scope;
   DGRM          Internal_Block_Diagram = its_IBD();
   REQ_Block     Funct_Req_dscr;
   REQ_Block     Nonfunct_Req_descr = NF_Req;
   REQ_Block     Abbreviations = Used_Abbreviations;  }
```

instance

Requirement Architect Tasks

inherit

Project File

```
REQ_Block   P1_ECU:ECU {
   MTHD          Black_Box_Interfaces = its_BB_IFs();
   MTHD          Event = its_Events();
   REQ_Block     System_Functions = P1_System_Functions;
   REQ_Block     Supplement_Functs = P1_SUP; }
```

Project File: Spec generation master file

```
SPEC          SRS_P1_ECU extends Spec_template {
 INFO         Scope= "This document is describing the
                      P1_ECU  as  White-Box";
 DGRM         Internal_Block_Diagram = its_IBD();
 REQ_Block    Funct_Req_dscr= P1_ECU:ECU;
 REQ_Block    Nonfunct_Req_descr = NF_Req;
 REQ_Block    Abbreviations = Used_Abbreviations; }
```

Legend:

red = project specific editing (RA Task)

italics = inherited

// grey = erased (RA Task)

VISION:EER

UC Features and Supplement Functions

```
REQ_Block   uC_Features {
   info              Frequency;
   info              Housing;
   info              NVM;
   info              RAM;
   info             (0..1)EEPROM;
   REQ_Block    (0..1)Dig_IO_init = DIO_init;
   REQ_Block    (0..1)ADC_init = xx_ADC_Block_PRTYs ;
   REQ_Block    (0..1)CAN_IF _init = CAN_Init;
   REQ_Block    (0..1)PWM_IF_init;
   REQ_Block    (0..1)TIC_IF_init;
   REQ_Block    (0..1)I2C_IF_init;
   REQ_Block    (0..1)SPI_IF_init;
   REQ_Block    (0..1)UART_IF_init ;
   REQ_Block    (0..1)Interrrupt_init; }
```

```
REQ_Block    SUP {
   INFO          Scope = "Overall supplement functions";
   REQ_Block   Mode_Handling = ECU_Modes;
   REQ_Block   UC_Features = xx_uC_Features;
   REQ_Block   Operating_System = AUTOSAR;
   REQ_Block   (1..*)Ext_Power_Supply = XX:Ext_POWER_SUPPLY;
   REQ_Block   (1..*)Int_Power_Supply;
   REQ_Block   (0..*)BUS_Handling;
   REQ_Block   (0..*)DIAG_Handling;
   REQ_Block   (0..*)NVM_Handler; }
```

```
Req_Block   Ext_POWER_SUPPLY {
  info             Scope = "Power Supply specific Requirements";
  req              Nominal_Voltage;
  req              Max_Voltage;
  req              Min_Voltage_for_full_ECU_operation;
  req              Max_Peak_Consumption;
  req              Availibilty;
  struc            Connector_pin;
  struc            GND_Connection;
  req              Disturbances;
 REQ_Block      (1..*)Power_Consumption_Mode = XX:Power_Mode; }
```

System Kit:

```
REQ_Block  DIO_Init  {
  info   Scope = "DIO Port Initializations";
  req    (1..*)Px.y_init; }
```

```
REQ_Block      ADC_Block_PRPTYs  {
  info   Scope = "ADC Block Initializations";
  req    Sampling_Frequency;
  req    Sampling_Mode;
  req    AD_Resulution;
  req    Reference_Voltage;   }
```

```
REQ_Block      CAN_Init  {
  info            Scope = "CAN Bus Initializations";
  req             BUS = HS_CAN_BUS;
  req             Buffer_size;
  req             Buffer_location;
  REQ_Block    Overflow_handling;   }
```

```
REQ_Block   AUTOSAR {
INFO          Scope = "Autosar specific Properties";
REQ_Block    (1..*)Service_Layer;
REQ_Block    (1..*)ECU_Abstraction_Layer;
REQ_Block    (1..*)Microcontroller_Abstraction_Layer;
REQ_Block    (0..*)Complex_Devices;
REQ_Block    (1..*)RTE;
REQ_Block    (1..*)Application_Layer; }
```

```
Req_Block   Power_Mode {
  info             Scope = "Power Mode specific Requirements";
  req              Max_current;
  req              Condition_for_Mode_Entry;
  req              Max_Mode_Readiness_Time;
  req             Activated_Functions; }
```

Mode Definitions

System Kit:

```
REQ_Block   ECU_Modes {
  MTHD        State_Diagram = its_State_diagram();
  REQ_Block   Life_Cycle_Phases = Life_cycle_phases;
  MTHD        System_Modes = Imported_system_modes();
  REQ_Block   SW_Modes = SW_modes; }
```

```
REQ_Block   Life_cycle_phases {
  MTHD        State_Diagram = its_State_diagram();
  REQ_Block   (1..*)ECU_production_,mode = xx_MODE;
  REQ_Block   (1..*)System_assembly_mode = xx_MODE;
  REQ_Block   (0..1)System_transport_mode = xx_MODE;
  REQ_Block   (1..*)System_operation_mode= xx_MODE;  }
```

```
REQ_Block   SW_modes  {
  MTHD        State_Diagram = its_State_diagram();
  REQ_Block   Start_up_mode= xx_MODE;
  REQ_Block   (1..*)Operation_mode= xx_MODE;
  REQ_Block   (0..*)Energy_safe_mode = xx_MODE;
  REQ_Block   (0..*)Fail_safe_mode = xx_MODE;
  REQ_Block   (0..*)Diagnostic_mode = xx_MODE;
  REQ_Block   Power_down_mode= xx_MODE; }
```

For any mode, the following items must be defined

```
REQ_Block   MODE {
  MTHD        Mode_Entry = its_Mode_entries();
  REQ         (1..n)Mode_Task;
  REQ_Block   (0..n)SubModes;
  MOD         (1..n)Mode_Exit = condition → exit_mode; }
```

Automatic creation/synchronization of state machine diagram

VISION:EER

Merging of System Functions

Autom. Generation

Conversion Result

```
REQ_Block   = P1_System_Functions
              merge P1_ECU and SFs {
   REQ_Block    SF = Measure_Switch_Status_SF;
   REQ_Block    SF = Evaluate_Signals_SF;  }
```

merge

System Kit

```
REQ_Block   SFs {
   REQ_Block     (0..*) SF = xx extends SF;  }
```

System Kit

```
Req_Block   SF {
  INFO          Scope;
  DGRM          Internal_Block_Diagram = its_IBD();
  REQ_Block     Interface_req = SF_its_IF_REQ;
  REQ           Execution_rate = its_Autosar_rate();
  REQ_Block     (0..*) IN_sig_conversion= xx: xx_CONV ;
  REQ_Block     (0..*) IN_msg_handling= xx: xx_BUS;
  REQ_Block     Signal_processing: xx:xx_SIGNAL_PROC;
  REQ_Block     (0..*) OUT_sig_conversion= xx: xx_IF ;
  REQ_Block     (0..*) OUT_msg_handling= xx: xx_BUS; }
```

Merging Result

```
Req_Block   merged_Measure_Switch_Status_SF {
  INFO          Scope;
  DGRM          Internal_Block_Diagram = its_IBD();
  REQ_Block     Interface_req = SF_its_IF_REQ;
  REQ           Execution_rate = its_Autosar_rate();
  REQ_Block     (0..*) IN_sig_conversion= xx: xx_CONV ;
  REQ_Block     (0..*) IN_msg_handling= xx: xx_BUS;
  REQ_Block     Signal_processing: xx:xx_SIGNAL_PROC;
  REQ_Block     (0..*) OUT_sig_conversion= xx: xx_IF ;
  REQ_Block     (0..*) OUT_msg_handling= xx: xx_BUS; }
```

Merging Result

```
Req_Block   merged_Evaluate_Signals_SF {
  INFO          Scope;
  DGRM          Internal_Block_Diagram = its_IBD();
  REQ_Block     Interface_req = SF_its_IF_REQ;
  REQ           Execution_freq;
  REQ_Block     (0..*) IN_sig_conversion= xx: xx_CONV ;
  REQ_Block     (0..*) IN_msg_handling= xx: xx_BUS;
  REQ_Block     Signal_processing = xx:xx_SIGNAL_PROC;
  REQ_Block     (0..*) OUT_sig_conversion= xx: xx_IF ;
  REQ_Block     (0..*) OUT_msg_handling= xx: xx_BUS; }
```

System Kit

```
Req_Block   SF_its_IF_REQ  {
  MTHD      Input_Signals = its_Input_signals();
  MTHD      Input_Frames = its_Input_frames();
  MTHD      Input_Messages = its_Input_messages();
  MTHD      Input_SW_IFs = its_Input_SW_IFs();
  MTHD      Output_Signals = its_Output_signals();
  MTHD      Output_Frames = its_Output_frames();
  MTHD      Output_Messages = its_Output_messages(),
  MTHD      Output_SW_IFs = its_Output_SW_IFs();
  MTHD      SF_Parameter = its_parameter();
  MTHD      SF_Events = its_Event_signals();
  MTHD      SF_Diag_Services = its_Diag_services(); }
```

inherit

inherit

Requirement Architect Tasks

Tailored Project File

```
REQ_Block   Measure_Switch_Status_SF extends
            merged_Measure_Switch_Status_SF {
  INFO          Scope = "Description of switch measurement";
  DGRM          Internal_Block_Diagram = its_IBD();
  REQ_Block     Interface_req = SF_its_IF_REQ;
  REQ           Execution_rate = its_Autosar_rate();
  REQ_Block     IN_sig_conversion= switch_signal:AI_CONV ->> SRS_P1_HSI;
  // REQ_Block     (0..*) IN_msg_handling= xx: xx_BUS;
  REQ_Block     Signal_processing = measured_switch_status:SIGNAL_PROC;
  // REQ_Block     (0..*) OUT_sig_conversion= xx: xx_IF ;
  // REQ_Block     (0..*) OUT_msg_handling= xx: xx_BUS; }
```

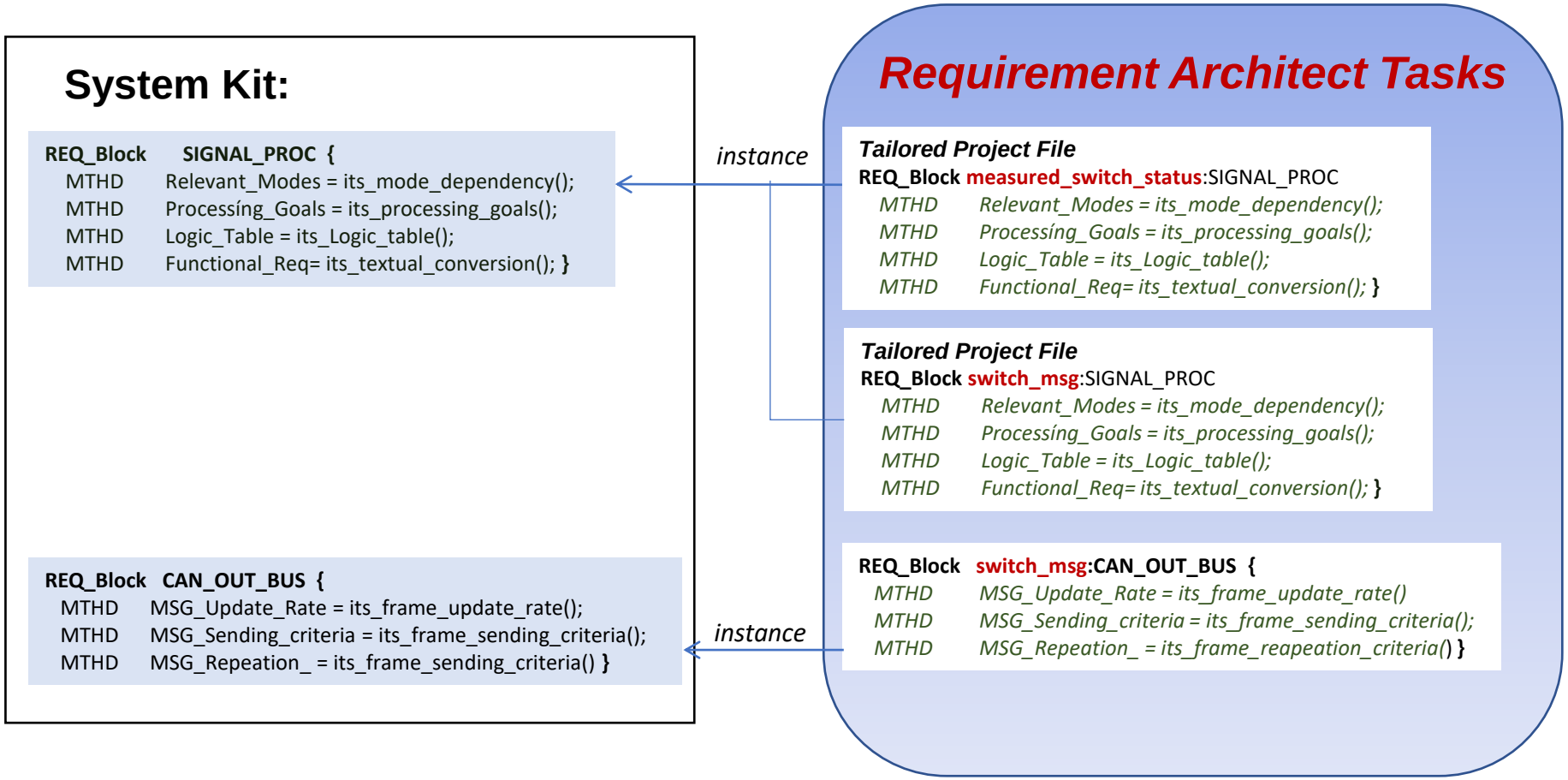
Described in separate Subspec

Tailored Project File

```
REQ_Block   Evaluate_Signals_SF extends
            merged_Evaluaute_Signals_SF {
  INFO          Scope = "Description of switch_msg evaluation";
  DGRM          Internal_Block_Diagram = its_IBD();
  REQ_Block     Interface_req = SF_its_IF_REQ;
  REQ           Execution_rate = its_Autosar_rate();
  // REQ_Block     (0..*) IN_sig_conversion= xx: xx_CONV ;
  // REQ_Block     (0..*) IN_msg_handling= xx: xx_BUS;
  REQ_Block     Signal_processing =  switch_msg:SIGNAL_PROC;
  // REQ_Block     (0..*) OUT_sig_conversion= xx: xx_IF ;
  REQ_Block     OUT_msg_handling= switch_msg: CAN_OUT_BUS; }
```

VISION:EER

SIGNAL PROCESSING



VISION:EER

Platform++ HW/SW Conversions → Spec SRS_P1_HSI

```
REQ_Block     CONV  {
  struc       IN_connect = xx_CONV_IN <-> xx_signal:IF_xx;
  struc       OUT_connect =xx_CONV_OUT<-> xx:xx
  INFO        Scope;
  DGRM        Conversion_Diagram = its_IBD();
  REQ_Block   (0..*)SubComponent;
  P_GOAL      (0..*)Processing_Goal; }
```

System Kit

inherit

```
REQ_Block     AI_CONV   extends CONV {
  struc       IN_connect = AI_CONV_IN ← xx_signal:AN_IN;
  struc       OUT_connect = AI_CONV_OUT → measured_xx_status:SIGNAL_PROC;
  INFO        Scope = "Conversion of analog signal into SW signal";
  DGRM        Conversion_Diagram = its_IBD();
  REQ_Block   SubComponent = AI_HW_CONV;
  REQ_Block   SubComponent = AI_SW_CONV;
  P_GOAL      Processing_Goal = "If Bat_voltage < 9V, then the previous signal
                                 value shall be used";
```

```
REQ_Block     AI_HW_CONV   extends CONV {
  struc       IN_connect = AI_HW_CONV_IN ← AI_CONV_IN;
  struc       OUT_connect = AI_HW_CONV_OUT →xx_PORT;
  INFO        Scope = "HW-conversion of analog input signal " ;
  DGRM        HW_Conv_Diagram = its_IDB();
  REQ_Block   SubComponent = AI_HW_RC_Filter;
  REQ_Block   SubComponent = AI_HW_Volt_Devider;; }
```

```
REQ_Block     AI_ SW_CONV extends CONV  {
  struc       IN_connect = AI_SW_CONV_IN ← xx_PORT;
  struc       OUT_connect = AI_SW_CONV_OUT → AI_CONV_OUT
measured_signal:SIGNAL_PROC;
  INFO        Scope = "SW conversion reqs of analog input signals";
  DGRM        SW_Conv_Diagram = its_IBD();
  REQ_Block   SubComponent = AI_ADC_IF;
  REQ_Block   SubComponent = AI_SW_Filter;
  REQ_Block   SubComponent = AI_DECODER;
  REQ_Block   SubComponent = AI_TEST; }
```

Plat form++

instance

Requirement Architect Tasks

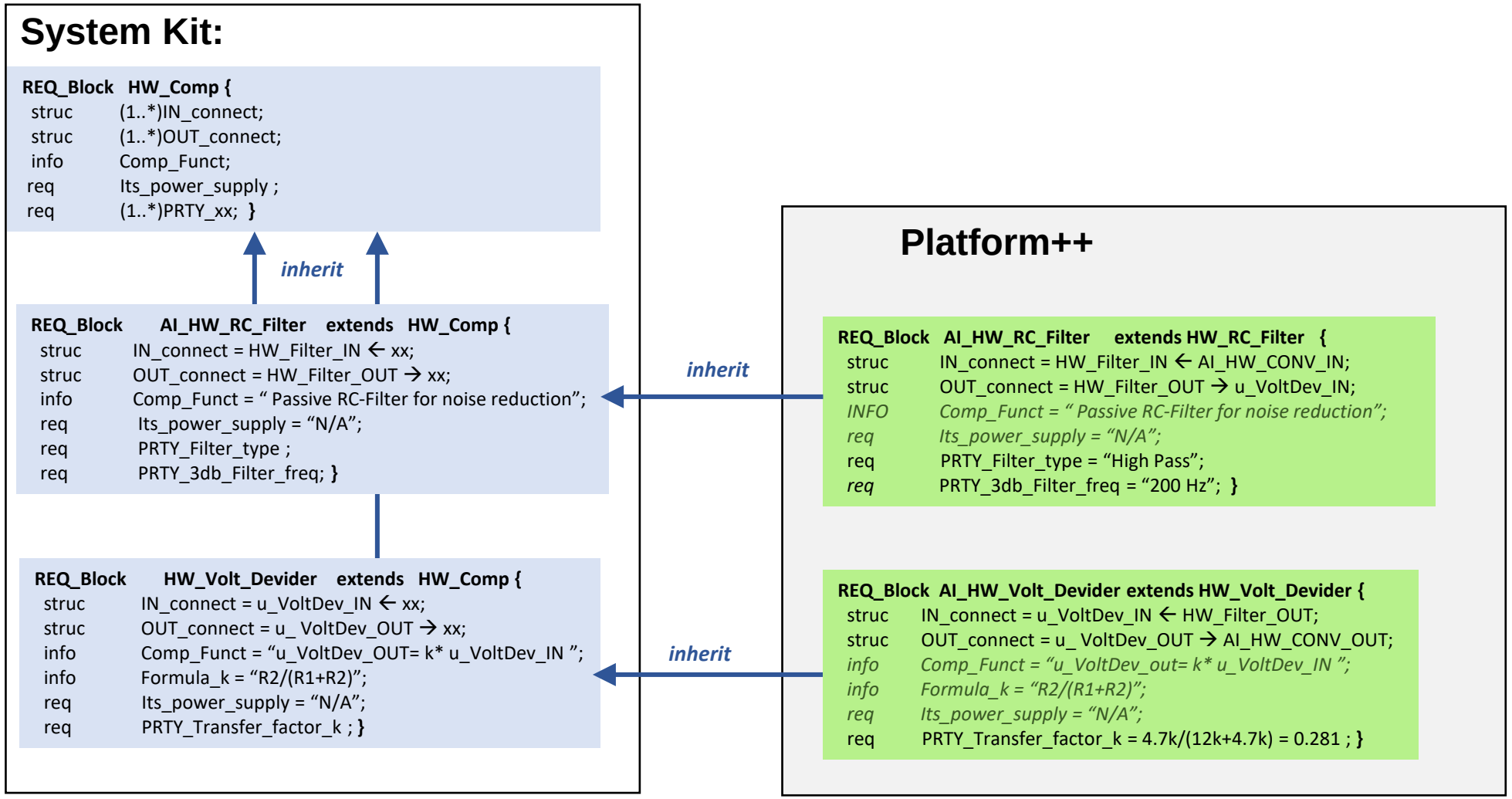
***Project File*: Spec generation master file**

```
SPEC          SRS_P1_HSI extends Spec_template {
  INFO        Scope= "This doc is describing the HW – SW Interface";
  DGRM        Internal_Block_Diagram = its_IBD();
  REQ_Block   Funct_Req_dscr= switch_signal:AI_CONV;
  REQ_Block   Nonfunct_Req_descr = NF_Req;
  REQ_Block   Abbreviations = Used_Abbreviations; }
```

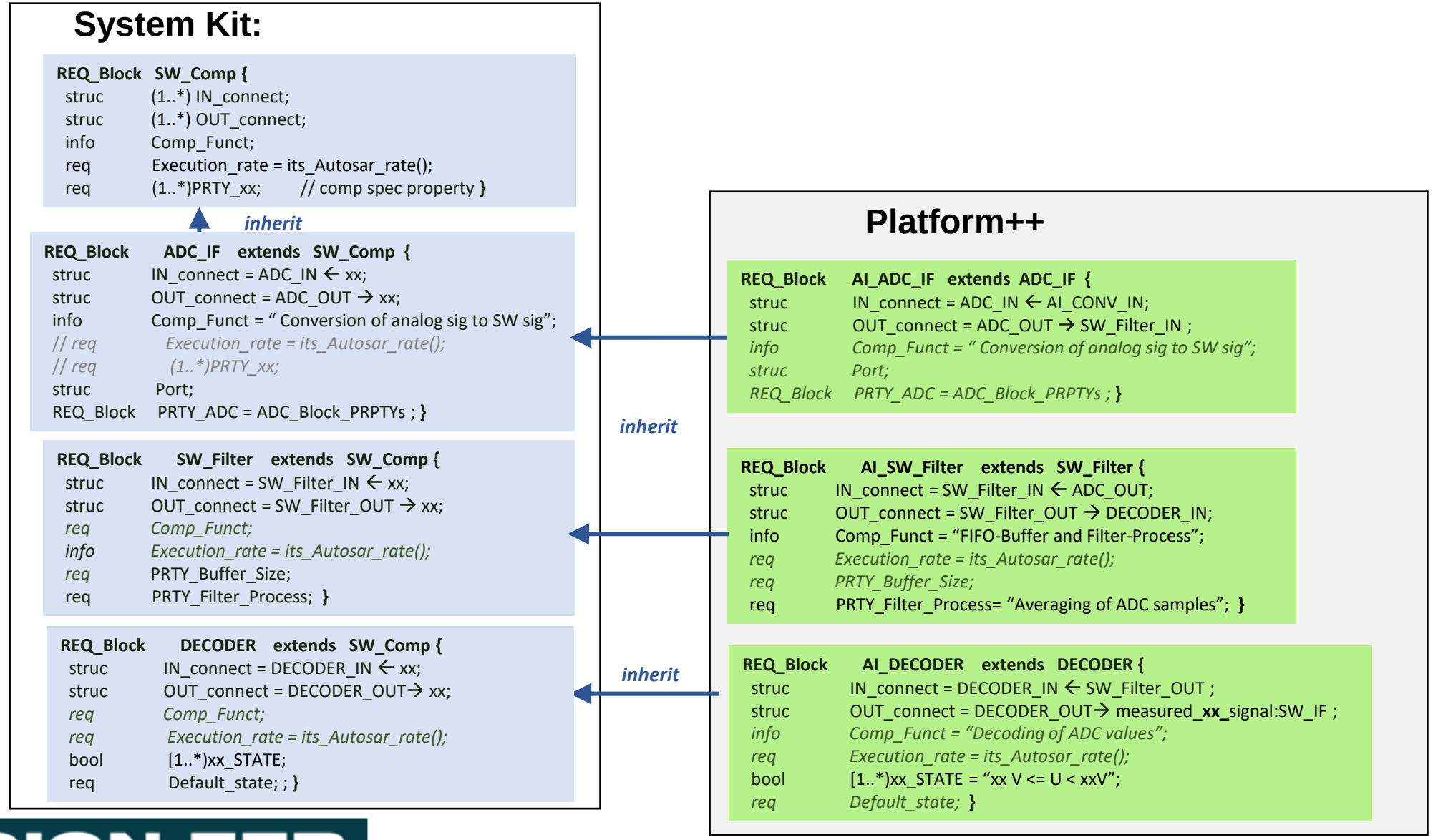
Tailored Project File

```
REQ_Block     switch_signal:AI_CONV  {
  struc       IN_connect = AI_CONV_IN ← switch_signal:AN_IN;
  struc       OUT_connect = AI_CONV_OUT →
                            measured_switch_status:SIGNAL_PROC;
  INFO        Scope= "Conversion of analog signal into SW signal";
  DGRM        Conversion_Diagram = its_IBD();
  REQ_Block   SubComponent = AI_HW_CONV;
  REQ_Block   SubComponent = AI_SW_CONV;
  P_GOAL      Processing_Goal = "If Bat_voltage < 9V, then the previous signal
                                 value shall be used; }
```

Platform++ HW Components



Platform++ SW Components



VISION:EER

Platform++ SW Components

System Kit:

```
REQ_Block     TEST    extends   SW_Comp {
 struc          IN_connect = TEST_IN ← xx;
 struc          OUT_connect = TEST_OUT→ xx;
 req            Execution_rate = its_Autosar_rate();
 info           Comp_Funct;
 REQ_Block    (0..*)PRTY_Test_Signal_Conversion;
 REQ_Block     (0..*)PRTY_DTC_Handling= xx:DTC;  }
```

```
REQ_Block      DTC  {
  struc       DTC_number;
  req         Err_Detect_Crit;
  req         DTC_Quali_Crit;
  req         DTC_Dequali_Crit;
  req         Error_Reaction;
  req         DTC_Indication;
  req         DTC_Delete_Protection;  }
```

Platform++

```
REQ_Block     AI_TEST    extends TEST {
  struc          IN_connect = TEST _IN ← DECODER_OUT;
  struc          OUT_connect = TEST_OUT → AI_SW_CONV_OUT;
  info           Comp_Funct = “Test of decoded AI signal values”;
  req            Execution_rate = its_Autosar_rate();
  // REQ_Block    (0..*)Test_Signal_Conversion;
  REQ_Block   (0..*) PRTY_DTC_Handling = xx:DTC;}
```

VISION:EER